****



Microcontrollers

Module 2: Programming, Controlling and Monitoring





PREPARED BY

**Academic Services Unit**

April 2012

© Applied Technology High Schools, 2012

Module 2:Programming, Controlling and Monitoring

**Module Objectives**

Upon successful completion of this module, students will beable to:

1. Define the purpose of indicator lights.
2. Assemble and test an LED circuit manually.
3. Connect an LED circuit to the BASIC Stamp and control it.
4. Write programs using DO…LOOP and FOR… NEXT loop commands.
5. Declare and use variables in a program.
6. Demonstrate the operation of a bi-color LED by assembling a simple ON/OFF circuit.
7. Control a bi-color LED with the BASIC Stamp.
8. Connect and test a pushbutton circuit manually.
9. Connect a pushbutton circuit to the BASIC Stamp and monitor its status.

**Module Contents:**

|  |  |  |
| --- | --- | --- |
|  | **Topic** | **Page No.** |
| 2.1 | Introduction | 2 |
| 2.2 | Building an LED Circuit | 3 |
| 2.3 | Variable Declaration | 7 |
| 2.4 | Loop Commands | 8 |
| 2.5 | Bi-color LED Control | 11 |
| 2.6 | Digital Input-Pushbuttons | 13 |
| 2.7 | Lab Activity 1 | 15 |
| 2.8 | Lab Activity 2 | 19 |
| 2.9 | Lab Activity 3 | 23 |
| 2.10 | Review Exercise | 26 |
| 2.11 | Assignment | 29 |

|  |
| --- |
| **2.1 Introduction** |
| In the previous module,you have learned the hardware parts of the BASIC Stamp. You have also got you familiar with the DEBUG and END commands. In this module, you will learn how to write simple programs to control indicator lights connected to the output pins of the BASIC stamp, and also how to monitor the status of the pins configured as inputs to the BASIC Stamp.  **Indicator Lights**  Indicator lights are so common that most people tend not to give much thought. Figure 2.1 shows the indicator lights on a laser printer. Depending on which light is ON, the person using the printer knows whether it is running properly or needs attention. A few examples of devices with indicator lights include car stereos, televisions, VCRs, disk drives, and alarm system control panels.    Figure 2.1 Examples of various devicesthat use microcontrollers  Turning an indicator light ON or OFF is a simple matter of connecting and disconnecting it from a power source. In some cases, the indicator light is connected directly to the battery or power supply, like the power indicator light on the Board of Education. Other indicator lights are switched ON and OFF by a microcontroller inside the device. These are usually status indicator lights which tell you what the device is up to.   |  | | --- | | **2.2 Building an LED Circuit** |   Most of the indicator lights you see on devices are called light emitting diodes or LEDs. An LED circuit can be connected to a BASIC Stamp and can be programmed to connect and disconnect power to the LED circuit. The BASIC Stamp can be programmed to do the following:   1. Turn an LED circuit ON and OFF at different rates. 2. Turn an LED circuit ON and OFF a certain number of times. 3. Control more than one LED circuit. 4. Control the color of bi-color LED circuit. |
| **2.2.1LED Circuit Components**  It is important to test components individually before assembling them. The LED circuit will consist of a resistor and an LED that will be assembled on a prototype board.  **Resistor:**A resistor shown in figure 2.2 is a component that ‘resists’ the flow of electricity or current. Each resistor has a value that informs how strongly it resists current flow. The resistance value is given in ohms.The colored stripes help you determine the value of the resistor, and the resistor used in this circuit has a value 47Ω. Review the color code that you have learned previously.    Figure 2.2: Resistor |
| **Light Emitting Diode:**  A diode is a one-way current valve, and a light emitting diode (LED) emits light when current passes through it. Unlike the color codes on a resistor, the color of the LED tells you what color it will glow when current passes through it. Since the LED is a one-way current valve, you have to connect it with proper polarity, or it will not work as intended. Figure 2.3 shows examples of LEDs. An LED has two terminals called the anode (+) and cathode (-). If you look closely at the LED’s plastic case, it’s mostly round, but there is a flat spot right near the shorter lead that tells you it is the cathode. An LED must be connected with a resistor in series with it to limit current flowing through its terminals.    Figure 2.3: Light Emitting Diodes  **PrototypingArea:**  You will build the circuit by plugging the LED and resistor leads into small holes called sockets on the prototypingarea as shown in figure 2.4. This prototyping area has black sockets along the top and along the left. The black sockets along the top have labels above them: Vdd, Vin, and Vss which are called the power terminals. They will be used to supply power to the circuit. The black sockets on the left have labels P0, P1, up to P15, and these are sockets used to connect your circuit to the BASIC Stamp module’s input/output pins. The white board is the solderless area called the breadboard where you will build circuits. |

|  |
| --- |
| Figure 2.4: Prototyping Area  Let us now review the following:   * 1. **Resistor Color Code**      1. If the color of the 1st band is yellow and the 2nd band is violet, what are the corresponding digits? The digits are \_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_.      2. If the color of the 3rd band is brown, how many number of zeros will you use? The number of zeros is \_\_\_\_\_\_\_\_\_\_.   2. **Light Emitting Diode**   An LED is a diode that emits light as the current passes through it from anode-to-cathode (forward direction). The LED is connected as follows:   * + 1. Anode: Connected to (+) side of voltage source. Typically has a longer lead.     2. Cathode: Connected to (–) side of voltage. Typically has a shorter lead.   Under normal working condition the LED voltage drops to 1.4V. The LED must be connected with a series resistor.   * 1. **Prototyping Area**  1. **White Breadboard:**  * Consists of two group of rows separated bya vertical channel. * Each row consists of 5 horizontal interconnected sockets. * There is no connection across channel (rows are not interconnected) * There is no connection between black of the headers to the white breadboard.  1. **Horizontal black strip is called Power Header and consists of:**   Vdd = +5 Volts.  Vss = GND = 0 Volts.  VIN = Supply Voltage from battery (9V) or wall DC supply.   1. **Side black strip is called I/O Header and consists of:**   16 I/O sockets labeled P0-to-P15⇒ I/O connections to the BS2. |
|  |

|  |
| --- |
| **2.3 Variable Declaration** |
| BASIC Stamp has four different types of variables, and each can store a different range of numbers as shown in the table below:   |  |  | | --- | --- | | Table 2.1: Variable Types and Values They Can Store | | | Variable Type | Range of Values | | Bit | 0 to 1 | | Nib | 0 to 15 | | Byte | 0 to 255 | | Word | 0 to 65535 |   You must explicitly allocate all of your variables before first use. Variable allocation lines generally should go at the beginning of your program.The **syntax** for variable declaration is as follows:  Name VAR VarType  **Rules for choosing Variable Names**   1. PBASIC keywords or labels cannot be used as variable names. 2. They can be up to 32 characters long. 3. They must start with a letter, and can contain a mixture of letters, numbers, and underscore (\_) characters. 4. PBASIC does not distinguish between upper and lower case, so the names MYVARIABLE, myVariable, and MyVaRiAbLe are all equivalent. 5. Select the smallest size of a VarType which can store the largest expected value of the variable. For example, if you need a VAR to hold the ON/OFF status (1/0) of switch, use a Bit. If you need a counter for a “FOR...NEXT” loop that counts from 1 to 100, use a Byte, and so on. |

|  |
| --- |
| **2.4 LOOP Commands** |
| Computer programs follow a sequential flow. They generally start at the 'top' of the program and progress line by line executing each instruction. If looping is introduced in a program, it will cause an instruction to branch, and repeat a set of instructions forever, or until a certain condition is met. PBASIC consists of several looping commands, some are listed below.   1. DO.....LOOP 2. DO.....LOOP UNTIL and DO WHILE … LOOP 3. FOR…… NEXT   **DO.....LOOP**  DO…. LOOP is a loop command that lets the program repeat a series of the instructions lying between them.  *Command syntax*:  DO  {........  Statement(s) ⇒ any valid PBASIC statements.  .........}  LOOP  *Explanation:*   1. Once the Program recognizes “DO”, it marks it as the loop starting point.The commands between the DO and LOOP keywords will execute over and over again.Once it recognizes the term LOOP, it will jump back to “DO”. 2. The steps that lie between them are executed forever, and therefore the loop is infinite.Even though this is called an infinite loop, the program can still be re-started by disconnecting and reconnecting power or pressing and releasing the Reset button.   **Conduct Lab Activity 1** |

|  |
| --- |
| **FOR … NEXT**  FOR … NEXT is another kind of looping command that can be used in theprogram to specify and monitor thenumber of loops. Thenumber of loops can beconfigured as an incrementing ordecrementing counter.  *Command syntax:*  FORVariable = XTOYSTEP  …..  …..  Next   * FOR: It marks the beginning of the loop * Variable: It is the variable you create to control the number of loops. * X (Start value): It will specify the initial value of the variable (0 to 65535). * Y (End value): It will specify the final value of the variable (0 to 65535). * STEP: Any number (0 to 65535) by which the variable is incremented or decremented in every loop. It is optional and equal to 1 by default in case it is omitted. * NEXT: It causes the counter to continue until the final value is reached.   *Example:*  Cars VAR WORD  FOR Cars=1 TO 10 ‘increment from 1 to 10 in a step of 1.  DEBUG ? Cars, CR  NEXT  Explanation:  Cars VAR WORD⇒ a word-sized variable is declared.  FOR: It marks the start of the loop; increment the initial value by 1 until the final value is reached.  1 TO 10: 1 is the initial value and 10 is the final value.  STEP: Ifomitted, X will increment by 1 after every loop until it reaches the final value 10.  DEBUG ?: Will display the value as “Cars = the current value of Cars”.  NEXT: If the counter value is less than 10, the program will branch or jump back to FOR. If the Cars value is 10, it will terminate and proceed to next instruction, if any. |

|  |
| --- |
| **2.5 Bi-Color LED Control** |
| The device shown in figure 2.5 is a security monitor for electronic keys. When an electronic key with the right code is used, the LED changes color, and a door opens. This kind of LED is called a bi-color LED.    Figure 2.5: Bi-color LED in a security device  The bi-color LED’s schematic symbol and part drawing are shown in figure 2.6.    Figure 2.6: Schematic symbol and Part drawing of bi-color LED |

|  |
| --- |
| The bi-color LED is just two LEDs in one package. Figure 2.7 shows how you can apply voltage in one direction and the LED will glow RED. By disconnecting the LED and plugging it back in reversed, the LED will glow green. As with other LEDs, if both terminals of the LED are connected to Vss, the LED will not emit light.    Figure 2.7: Bi-color LED and applied voltage  **Conduct Lab Activity 2** |

|  |
| --- |
| **2.6 Digital Input-Pushbuttons** |
| There are many devices with pushbuttons that we use in our daily life, such as a computer, mouse, calculator, microwave, handheld remote, VCRand so on. In each device, there is a microcontroller scanning the pushbuttons and waiting for the circuit to change. When the circuit changes, the microcontroller detects the change and takes action.  In the previous section, you programmed the BASIC Stamp to send high and low signals, and you used LED circuits to display these signals. Sending high and low signals means you used a BASIC Stamp I/O pin as an output. In this section, you will use a BASIC Stamp I/O pin as an input, and you will program the BASIC Stamp to recognize whether the pushbutton is pressed or not.  Figure 2.8 shows the schematic symbol and the part drawing of a normally open pushbutton. Two of the pushbutton’s pins are connected to each terminal. This means that connecting a wire of part lead to pin 1 of the pushbutton is the same as connecting it to pin 4. The same rule applies with pins 2 and 3. The reason the pushbutton does not have just two pins is because it needs stability.    Figure 2.8: NO pushbutton, part drawing and schematic symbol |

|  |  |
| --- | --- |
| **Pushbutton Operation**  Figures (2.9a & 2.9b) illustrate the operation of the PB. When the PB is not pressed, there is no connection between the 1,4 and 2,3 terminals(open circuit state). When the button is pressed, the gap between the 1,4 and 2,3 terminals is bridged by a conductive metal (closed circuit state). | |
| a: Not pressed | b: Pressed |
| Figure 2.9: NO Pushbutton States | |
| **Conduct Lab Activity 3** | |

|  |  |
| --- | --- |
| **2.7Lab Activity 1** | |
| **Objective:** To build and program an LED circuit. | |
| **Background**: In this experiment you will build an electronic circuit on the BoE. The circuit consists of an LED in series with a 470Ω resistor. In this circuit the following two tests will be conducted:  **Task#1:** Testing the LED without connecting the circuit to the BS2.  **Task#2:** Testing the LED by connecting the circuit to the BS2. | |
| **Task #1: LED Test Circuit without BASIC Stamp** | |
|  |  |
| Figure 2.10a: LED Circuit on breadboard | Figure 2.10b:Circuit Diagram |
| **Procedure:**   1. Disconnect power from your BoE by setting the 3-position SW on the BoE to position-0. 2. Refer to Figure 2.10a and 2.10b and make the following connections: 3. Connect the cathode terminal of the LED to Vss. 4. Connect the anode terminal into a suitable socket. 5. Connect one terminal of R (470Ω) to the anode of the LED. 6. Connect the other terminal of R (40kΩ) to VDD. 7. Turn on the Power of the BOE. What do you observe?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| **Task #2: LED Test Circuit with BASIC Stamp** | |
| Figure 2.11: LED circuit with BASIC Stamp   1. Disconnect power from your BoE. 2. Modify the circuit to match the one in figure 2.11. 3. Enter the following program on the BASIC Stamp editor and save it as **LEDOnOff.bs2**   **‘{$STAMP BS2}**  **‘{$PBASIC 2.5}**  DEBUG“The LED connected to Pin14 is blinking”  DO‘The loop starts here  HIGH 14‘Set P14 High  PAUSE 500 ‘Wait for 0.5 Sec  LOW 14‘Set P14 Low  PAUSE 500‘Wait for 0.5 Sec  LOOP‘Jump back to DO  END   1. Download and run the program. 2. Verify that the LED flashes ON and OFF once per second. | |
| **How LEDOnOff.bs2 Works**  The command DEBUG“The LED connected to Pin14 is blinking”makes this statement appear in the Debug terminal. The command HIGH 14 causes the BASIC Stamp to internally connect I/O pin P14 to Vdd. This turns the LED ON.  The command PAUSE 500 causes the BASIC Stamp to pause or stop for 0.5 second while the LED stays ON. The number 500 tells the BASIC Stamp to wait for 500/1000 of a second. The command syntax is **PAUSE duration** where the duration is in milliseconds. | |

|  |
| --- |
| The command LOW 14 causes the BASIC Stamp to internally connect I/O pin P14 to Vss. This turns the LED OFF. Since LOW 14 is followed by another PAUSE 500, the LED stays OFF for 0.5 second.  The reason the code repeats itself, or the LED keeps blinking over and over again is because it is nested between the PBASIC keywords DO and LOOP. The four commands between the DO and LOOPare executed by the BASIC Stamp over and over again. Therefore, the LED will keep flashing until the power is disconnected, or until the battery runs out. |
|  |

|  |
| --- |
| **2.8Lab Activity2** |
| **Objective:** To build and test the Bi-color LED Circuit |
| **Background:** Initially, you will conduct a manual test for the bi-color LED. The bi-color LED will be connected to the BASIC Stamp in the second test. Controlling a bi-color LED with the BASIC Stamp requires two I/O pins. After you have manually verified that the bi-color LED works using the manual test, you can proceed with the second test.  **Part #1: Manual Test**   1. Disconnect power from your Board of Education. 2. Build the circuit shown on the left side of Figure 2.12. 3. Reconnect power and verify that the bi-color LED is emitting red light. 4. Disconnect power again. 5. Modify your circuit so that it matches the right side of Figure 2.12. 6. Reconnect power. 7. Verify that the bi-color LED is now emitting green light. 8. Disconnect power. |
| Figure 2.12: Manual Bi-color LED Test |
| **Part #2: Controlling bi-color LED with BASIC Stamp**   1. Connect the bi-color LED circuit to the BASIC Stamp as shown in figure 2.13.     Figure 2.13: Bi-color LED connected to BASIC Stamp  Figure 2.14 shows how you can use P15 and P14 to control the current flow in the bi-color LED circuit.    Figure 2.14: Current through RED (above) and Green (below) LEDs.   1. Enter the TestBiColorLed.bs2 codeinto the BASIC Stamp editor. 2. Run the program, and verify that the bi-color LED cycles through red, green and off states.     **Part #3: Introducing a counter to limit the LED cycles.**  A variable named counter is used to control the number of times the LED blinks.   1. Rename and save TestBiColorLed.bs2 as TestBiColorLedCount.bs2. 2. Add a counter variable declaration before the DO statement:   Counter VAR BYTE   1. Nest the FOR….NEXT loop below within the DO…LOOP. |

|  |
| --- |
| Your program should like this:  Counter VAR BYTE  DO  FOR counter = 1 to 50  HIGH 15  LOW 14  PAUSE counter  LOW 15  HIGH 14  PAUSE counter  NEXT  LOOP   1. Run the program and count the number of times the LED blinks/flashes.   **Observation:**  The number of times the LED flashes is \_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |  |
| --- | --- |
| **2.9 Lab Activity 3** | |
| **Objective:**  To build and test a pushbutton circuit. | |
| **Background:**  Initially, you will conduct a manual test for the pushbutton. The pushbutton circuit will be connected to the BASIC Stamp in the second test. BASIC Stamp program will monitor the status of the PB (ON/OFF) and display its value as 1 or 0.  **Part #1: Manual Test**   1. Build the circuit shown in figure 2. Verify that the LED in your test circuit is OFF. 3. Press the pushbutton, and check the LED status. When the pushbutton is pressed, the LED must turn ON. 4. Press and hold the pushbutton, and verify that the LED emits light while you are holding the pushbutton down. | |
|  |  |
| Figure 2.15a: Pushbutton Circuit Connections | Figure 2.15b: Pushbutton Circuit Schematic |

|  |  |
| --- | --- |
| **Observations:**   1. When the PB is not pressed, the state of the LED is....................... 2. When the PB is pressed, the state of the LED is............................ 3. When you press and hold down the PB, the state of the LED is …………   **Part #2: Pushbutton Circuit with BASIC Stamp**  Refer to figure 2.16. When the PB is pressed, P3 will be connected to Vdd via 220Ω resistor. The BS2 will detect the state of P3 as High and set its value to 1. When the PB is not pressed, the I/O P3 is pulled down to Vss via 220Ω and 10kΩ resistors. The BS2 will detect the state of P3 as Low and set its value to 0.The command that could be used to read the status of I/O pin is IN. The command syntax is **IN**pin, for example, the command IN3 will read the status of pin 3.  **Procedure:**   1. Turn off the power to the BoE. 2. Build the circuit shown in figure 3. Switch ON the power to the BoE. | |
| Figure 2.16a: BS2 monitoring I/O P3 | Figure 2.16b:Schematic diagram |

|  |
| --- |
| 1. Enter the following program code (ReadPushbuttonState.bs2):   ‘What’s a Microcontroller-ReadPushbuttonState.bs2  ‘{$STAMP BS2}  ‘{$PBASIC 2.5}  DO  DEBUG ? IN3  PAUSE 250  LOOP  END   1. Run the program. 2. Verify that the debug terminal displays the value 0 when the pushbutton is not pressed. 3. Verify that the debug terminal displays the value 1 when the pushbutton is pressed and held.   **Observations:**   1. When the PB is not pressed, the value displayed is....................... 2. When the PB is pressed and held, the value displayed is................... |
|  |

|  |
| --- |
| **2.10 Review Exercise** |
| 1. How long would PAUSE 1000 last?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. What program code would you use to cause the BASIC Stamp to do nothing for an entire minute?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. What are the different types of variables used by BASIC Stamp?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. Can a byte hold the value 500?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. What will the command HIGH 7 do?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |  |
| --- | --- |
| 1. Refer to the figure below and answer the following questions: | |
|  | **‘{$STAMP BS2}**  **‘{$PBASIC 2.5}**  DEBUG“The LED is blinking”  DO  HIGH 14  PAUSE 500  LOW 14  PAUSE 500  LOOP  END |
| 1. Draw the schematic diagram for the LED circuit. 2. Modify the LedOnOff.bs2 program to make the LED flash ON and OFF four times in one second.   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| 1. Rewrite the program to make the LED circuit flash ON and OFF 500 times before it stops.   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| 1. What happens between the terminals of a normally open pushbutton when you press it?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| 1. What is the value of IN3 when a pushbutton connects it to Vss?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| 1. What does the command DEBUG ? IN3 do?   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

|  |
| --- |
| 1. Write a BS2 program to read the state of the pushbutton in the circuit shown below:     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |
| --- |
| **2.11 Assignment** |
| Make a 10-second countdown using one yellow LED and one bi-color LED. Make the bi-color LED start out red for 3 seconds. After 3 seconds, change the bi-color LED to green. When the bi-color LED changes to green, flash the yellow LED ON and OFF every second for 10 seconds. The program must repeat forever till the reset button is pressed or until the power to the BASIC Stamp is switched OFF. |